Comprehensive Large Array-data Stewardship System (CLASS)

Software Development Guide

(Version 1)

October 1, 2002

Revisions

Version	Description of Version	Date Completed
Draft V1	Initial draft	7/22/02
1.0	Incorporated CPMT input – approved by CPMT	10/01/02

Review & Approval

Software Development Guide Review History

Reviewer	Version Reviewed	Signature	Date
Constantino Cremidis/CSC			
Alexander Kidd/OSDPD			
Geof Goodrum/NCDC			
Carlos Martinez/Marada Corporation			
Ted Habermann/NGDC			
Eric Kihn/NGDC			
David Vercelli/NESDIS			
Robert Mairs/NESDIS			
Anita Konzak/CSC/QMO			

Table of Contents

Table of Figures

1 Introduction

This software development guide describes the standards and procedures to be followed throughout the development of the Comprehensive Large Array-data Stewardship System (CLASS). The CLASS development project includes several separate development groups from different organizations and different geographic areas. To ensure consistent quality and compatibility of the various components of CLASS developed by this distributed team, all members of the CLASS development team will follow the standards and procedures defined here. The CLASS Quality Management (QM) personnel will provide oversight and guidance for all development groups in the application of the CLASS processes, as defined in the CLASS QM Plan.

This section of the guide describes the overall development environment for CLASS, including organization, technology, and baseline documents. Subsequent sections describe the processes, standards, and procedures for release planning, development (detailed design and coding), testing, and documentation.

This document will be updated throughout the development life of CLASS to incorporate lessons learned and process improvements. The CLASS Project Management Team (CPMT) must approve any updates to this document. Once approved, updates will be distributed to all members of the CLASS development team and posted in the CLASS online library.

1.1 General development approach

The overall goal for CLASS is to provide one place for access to all NOAA/NESDIS data. Specifically, CLASS is currently scheduled to support data archiving and retrieval for seven major campaigns over the next several years:

- NOAA and Department of Defense (DoD) Polar-orbiting Operational Environmental Satellites (POES)
- NOAA Geostationary-orbiting Operational Environmental Satellites (GOES)
- National Polar-orbiting Operational Environmental Satellite System (NPOESS)
- The NPOESS Preparatory Program (NPP)
- National Aeronautics and Space Administration (NASA) Earth Observing System (EOS)
- NOAA NEXt generation weather RADAR (NEXRAD) Program
- European Meteorological Operational Satellite (Metop) Program

To meet these goals in a cost-efficient manner, CLASS is being developed in an evolutionary manner, re-using existing system functionality as possible. The data archive and distribution functionality is based on the Satellite Active Archive (SAA) system, currently supporting NOAA Polar-orbiting Operational Environmental Satellites.

The overall methodology used in the development of CLASS is iterative and release-based. The iterative approach allows for the continued refinement of detailed requirements and design as new campaign requirements are defined. Implementation is release-based in order to minimize risk to the operational baseline as the system evolves.

Figure 1 shows the high-level process flow for the release life-cycle.

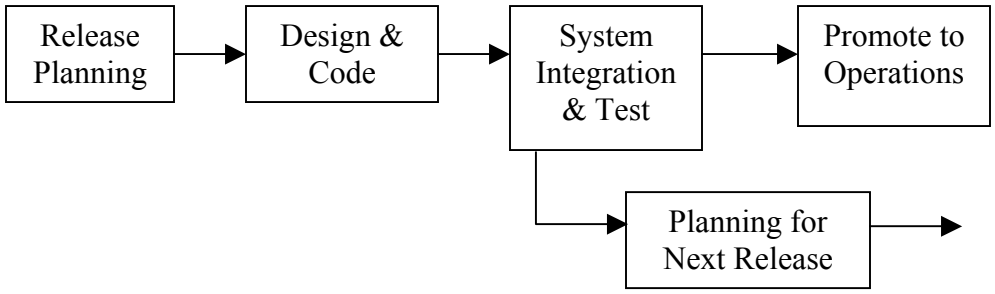


Figure 1 - Release life-cycle

1.2 Project organization

The CLASS project is distributed across several development groups that are organizationally and geographically distinct. The project is managed by a joint project management team (the CPMT) and by a common set of baseline documents, standards, and processes. The CLASS System Engineering Team (SET) includes representatives from each development group and coordinates the technical direction for CLASS. Figure 2 shows the project organization. The charter for the CPMT, and the roles and responsibilities of key team members, are defined in the CLASS Master Project Management Plan.

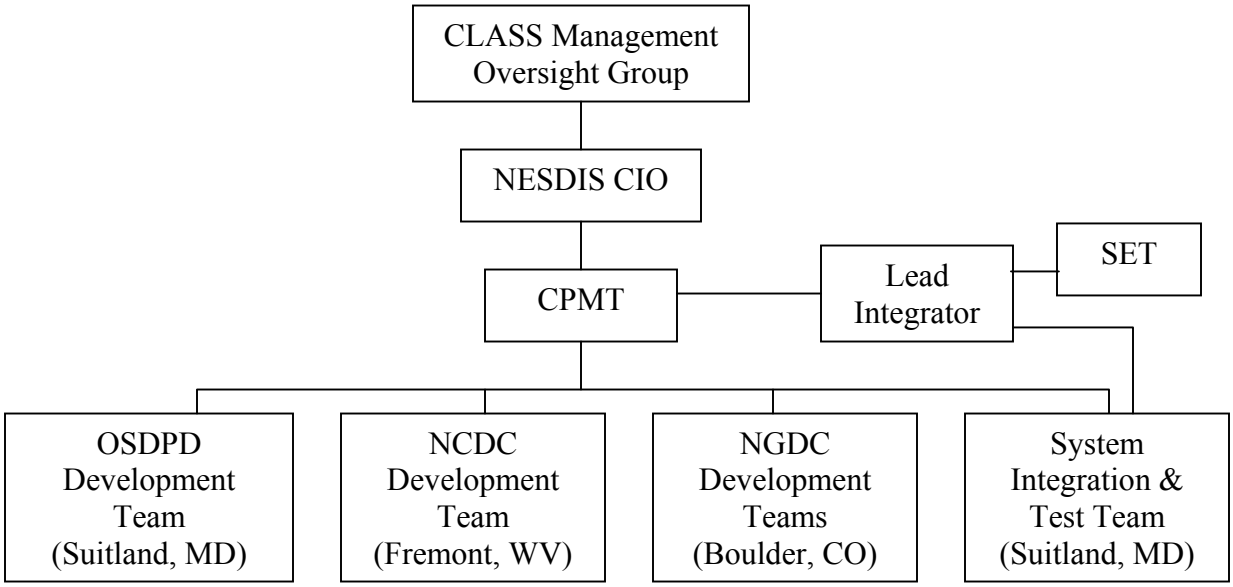


Figure 2 - CLASS Project Organization

1.3 Development environment

Each development group maintains its own local development environment for coding and unit and component integration testing. A common source control system is maintained in the system integration and test environment, which provides the baseline code for CLASS. When a component has been developed and tested locally, it is then turned over to the system integration and test team for integration into the CLASS test environment. Figure 3 shows the overall flow of software changes within the CLASS project. Details of each step are defined in later sections of this guide.

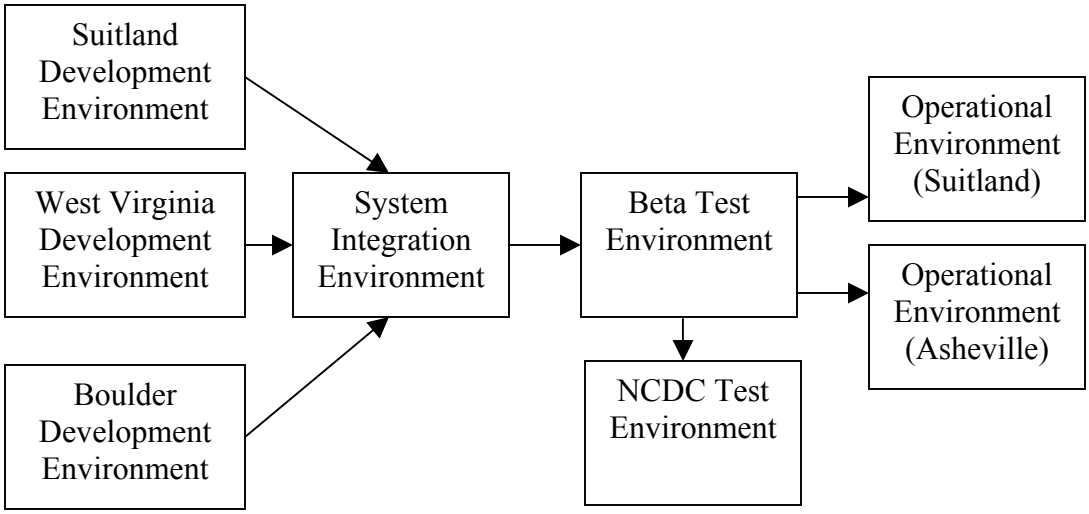


Figure 3 - CLASS Technical Environments

1.4 Related documents

All CLASS baseline documents are stored in the CLASS online library, available in the CLASS Project Development Team space in NOAAForge (class1.nesdis-hq.noaa.gov). These include the following documents:

- CLASS Configuration Management Plan
- CLASS Quality Management Plan
- CLASS Master Project Management Plan
- CLASS Test plan
- Design documentation
- Software Description documentation
- Software Standards for Information Processing Division (IPD), June 30, 2001

2 Release planning

The success of the release-based approach is largely determined during the release planning effort. The CPMT must allocate requirements to each release in a manner that

- Addresses campaign needs and schedules
- Groups related design and code changes so as to minimize code disturbance
- Provides for maximum generalization of functionality

The major activities that take place during release planning are described in this section. These activities may be iterative during the course of defining a release: an initial set of requirements is allocated to the release, the effort for implementation is estimated, and the resulting schedule for delivery is determined. If the resulting schedule is not

acceptable, the requirements allocation is revisited to change the scope of the release to one that can be completed in the required timeframe, or the resources allocated to the release are increased to meet the target completion date.

2.1 Scope definition

CLASS system and allocated requirements are maintained in the DOORS requirements management tool. Associated with each allocated requirement are the source (campaign or other source) and the required operational date. The CLASS CM plan describes the process for managing new or proposed requirements.

Near the end of development of a release, the CPMT, SET, and Lead Integrator begin assessment of the scope for the next release:

- Existing Level III configuration change requests (CCRs) are reviewed to verify the allocation to the next release.
- Requirements previously allocated to the next release time period are reviewed to determine if the release allocation is still desirable.
    - Related requirements that comprise a single functional capability are grouped together. Each group should include only requirements that are so tightly coupled that it is not reasonable to implement one without concurrently implementing all in the group.
    - All requirements that have not been implemented yet are reviewed to determine if any are logically related to those groups planned for this release. If any are identified, they are added to the appropriate grouping.
    - A Level III CCR is created for each functional change, and entered into the CCR tool. All changes are documented in CCRs before implementation begins to facilitate tracking of the change status.

The release scope includes the CCRs defining new capabilities to be implemented and existing problems that will be corrected in the release. The SET constructs the initial release list, which is then reviewed by the CPMT for consistency with current priorities. The list is then assessed for effort and schedule, as described in the following sections, and revisited as necessary. When all parties – development groups, test team, and management – are satisfied that the allocated work can be completed in the desired timeframe, with acceptable quality, the release scope is documented by the list of allocated CCRs. The CPMT has final authority on release content.

Any changes to scope (i.e., inclusion of additional CCRs or elimination of CCRs) proposed by the development groups during the release implementation must be reviewed by the integration test team and approved by the CPMT.

2.2 Effort estimate

In order to meet the CLASS schedule and quality commitments, accurate effort estimates are a key input to the release planning activity. The accuracy of software estimates is driven by two key considerations:

- The level of detail of understanding of the requirement to be implemented or problem to be corrected, and
- The historical organizational experience with similar implementation efforts (similar in application, platform, programming language, etc.)

The level of detail of understanding will change during the system life cycle. Initial effort estimates will be less accurate than those derived after detailed design is completed. Estimates will be reviewed at least twice during a release, and more frequently as warranted: once at initial release planning, and once after the developer has completed requirements analysis and design. The initial release plan should include schedule contingency based on expected changes in the effort estimate after design. If changes in estimates later in the release suggest that the schedule will not be met, the CPMT will review the release scope and schedule to determine if the schedule should be adjusted, the scope revised, or additional resources applied.

The historical experience for CLASS is strong, since the system is based on existing systems and supported by experienced developers. In some cases, the new capability may be sufficiently unlike existing functionality, or the group assigned to that new capability new to the project, so that the estimate is less certain than other parts of the system. These areas should be identified during the estimation process and monitored closely during development for deviation from the estimates. When necessary, outside experts may be consulted to provide the experience needed for a reliable estimate.

The lead integrator prepares a preliminary rough estimate for each CCR when it is received. Each development group is responsible for reviewing and refining the estimates for CCRs assigned to their group, during the release planning phase. The estimation process used for CLASS includes the following major activities, to be performed for each CCR.

- Partition the requirement or problem into the lowest level objects or functions possible. These may be fairly high-level at the initial estimate, and more detailed for later re-estimates.
- For each type of object, identify the “estimating unit” (e.g., modules, lines of source code (SLOC), user interface screens).
- Estimate the number of units for each object or function, based on previous releases or similar projects.
- Estimate the complexity (high, medium, low) for each object or function.
- Define the expected productivity (hours per unit) for each level of complexity, based on previous releases or similar projects.
- Compute the effort estimate for each object or function, based on the expected productivity and units estimate for that object or function.
- Sum the effort estimates for the objects or functions for each CCR.
- Sum the effort estimates for the release.

The estimator completes a worksheet for each CCR. Figure 4 shows a sample worksheet: productivity numbers are examples only; actual numbers will be defined by the software

development leads and refined during the life of the CLASS project based on actual experience. Appendix C contains a blank worksheet.

CLASS Estimation Worksheet

Estimating Unit Characteristics

ID	Estimating Unit	Complexity Productivity Estimate (hours/unit)		
		Low	Medium	High
FN	Function	8	40	120
SC	UI Screen	4	24	40
SC OB	UI Screen Object	.5	2	4
DSI	Delivered Source Instruction	.5	1	2

Work Estimate

Partition ID	Estimating Unit ID	Complexity	Size	Effort (Hours)
New capability A	FN	M	3	120
New capability A UI	SC	H	1	40
New capability A UI	SC	L	5	20
Modify screen B	SC OB	M	3	6
New function C	DSI	M	200	200
Total Effort Estimate				386

Figure 4 - Sample Estimation Worksheet

The following detailed characteristics should be considered in preparing the estimate.

- Decompose requirements into objects or functions.
- Identify like objects/functions and group into “develop once” category.

- Identify Non-Developmental Item (NDI) or reusable objects/functions.
- Identify Activity Controller Path definitions or system configuration
- Estimate code size for each object/function to be developed.
- Estimate adaptation code for adjusting NDI or reusable code, if appropriate.
- Estimate initialization, termination, and error handling code for each process or partial process.
- Estimate new data definitions needed to support each process or partial process.
- Estimate screen/form handling logic and definition statements.
- If any tool needs to be developed (e.g., test drivers, development tools), estimate code size for each tool.

The SET supports the development groups in preparation of the estimates and reviews the estimates on completion.

2.3 Release schedule

Each Technical Area Lead (TAL) prepares a detailed MSProject plan for their part of the release, based on input from the development group. This plan includes all tasks, milestones for each task, resources assigned to each task, and dependencies between tasks. The tasks should be categorized according to the CLASS Work Breakdown Structure (WBS), and the work defined using a standard Earned Value Method, as documented in the CLASS Project Management Plan.

The following types of development activities should be included in the release plan:

- Initial analysis, design, code, and development testing
- Review of design, code, test plans, test results, and documentation
- Rework, for problems found during the integration and test phase
- Documentation

Integration and system test tasks include

- Test planning
- System builds
- Test execution
- Retesting of software where problems were found in initial testing
- Documentation of test results

If there are dependencies on the order in which new capabilities are tested, these must be defined in the release plan.

Plans should also include allocation of effort (both development and system testing) for analysis of new CCRs that are received during the release period.

The CLASS project control office rolls up the individual group plans into one master release plan, working with the CPMT to resolve any conflicts encountered. The CPMT reviews and approves the final release schedule.

2.4 Metrics collection

At the start of each release, each group completes a process database measurement form. This form includes project description information and effort estimates for the release activities. At the end of the release, actual data related to the release size, effort, and quality are recorded. These metrics provide a historical baseline for use in planning future releases. Attachment A shows the measurement form for data collection.

Each TAL provides input to the Release Notebook for each release that contains the following:

- The planned release scope (list of CCRs)
- The release estimates, including the basis for estimates
- The initial release plan
- Release actuals (collected during and at the end of the release):
    - Scope (CCRs included in the delivery)
    - Effort
    - Schedule
    - Quality measures (see Attachment A, and Sections 3 and 5)
- Lessons learned and recommendations for future releases

3 Software development process

After the scope of the release has been defined and approved by the CPMT, each CCR is assigned to one or more developers for implementation. Development activities include

- Requirements analysis to ensure that the requirement or problem is understood
- Detailed design to define the implementation approach
- Coding of the new or changed functionality
- Unit and component testing of the new or changed code
- Peer review

After the development group has tested the function, it is turned over to the CMO for promotion to the integration and test environment, for integration into CLASS and formal testing. Independent integration and testing activities are addressed in Section 5. This section describes the process and procedures for the development activities. Coding standards are defined in Section 4.

3.1 Process overview

In order to identify and correct problems as early in the development process as possible, the CLASS project will use a series of peer reviews as each function is implemented. Figure 5 shows the process flow for development and the points where peer review is conducted. This section describes the steps in the process. Attachment B describes in detail the procedure for peer reviews, describing each different type of review. The checklists for each review are included in Section 3.2.

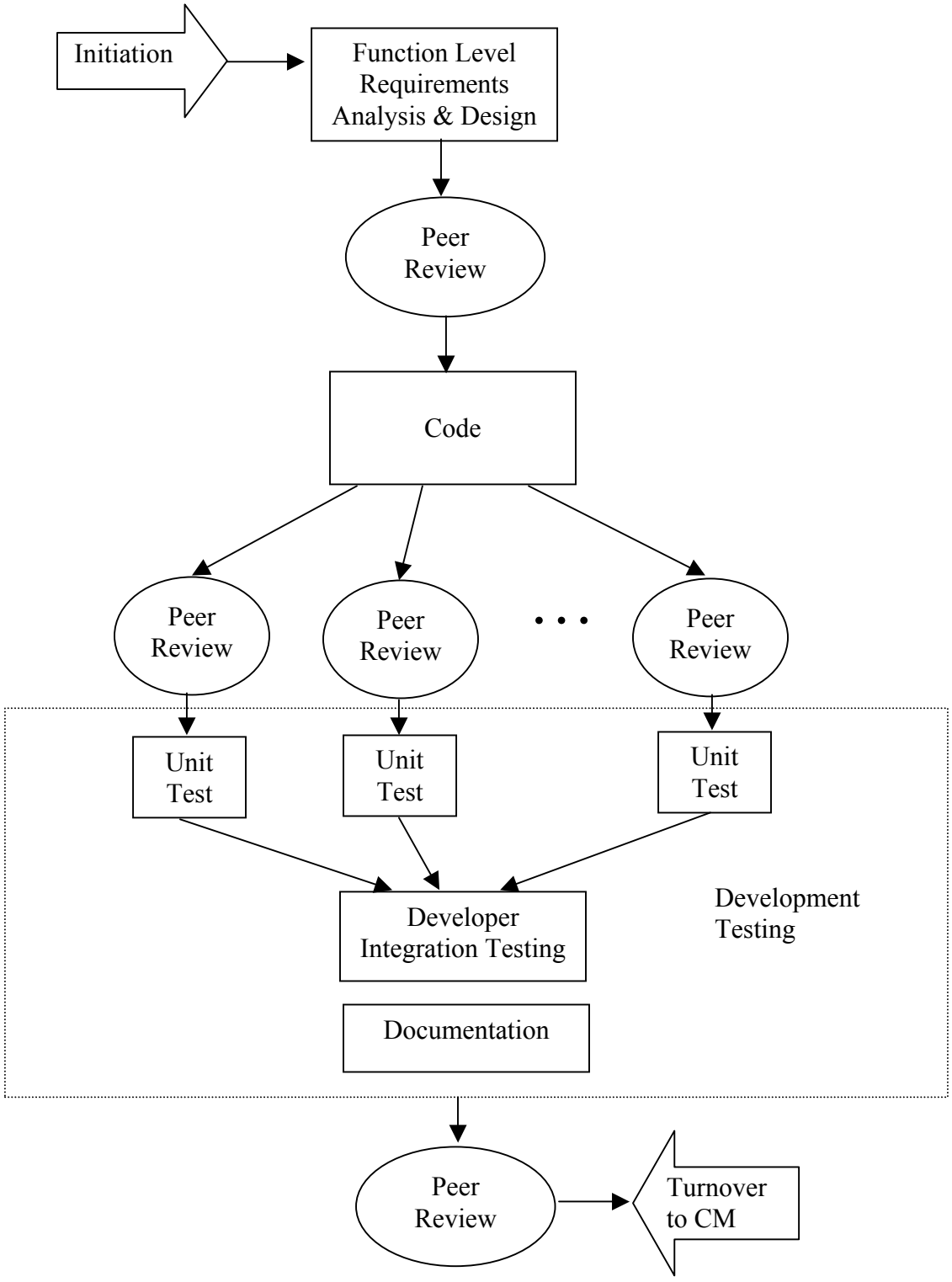


Figure 5 - Development Process

Initiation

The process is initiated when the software development lead assigns a CCR to a development team, which includes at least two members to facilitate peer review. Typically, one member will do all the development for a particular CCR, and the other will perform the peer review. Based on the complexity of the CCR, the software development lead will designate the level of peer review required for that CCR, and identify any additional peer reviewers, if necessary. For large, complex, or critical CCRs, a formal Work Product Inspection (WPI) may be required at the completion of design, and group reviews required for code and test. For minor changes to the code (e.g., correcting an error in a single line of code), only a one-on-one review at the completion of developer testing may be required.

When the CCR is assigned to a development team, the development lead changes the CCR Status to Assigned and the Supervisor to the lead assignee.

Function Level Requirements Analysis and Design

When the assigned developer begins work on the CCR analysis, the developer changes the CCR Status to Work in Progress. The developer reviews the CCR and gets clarification on any requirements. The developer then develops a detailed design for implementation, including identifying any new or modified code, user interface changes, and other interface changes. The developer re-estimates the effort required for implementation. If the effort estimate is significantly higher than the original estimate developed during release planning, the developer notifies the software development lead, so that the CPMT can determine if the effort should remain included in the current release and if additional resources are required.

When the developer has completed analysis of the requirements and design, and is ready to begin coding, the developer schedules the design peer review, as defined at initiation. The peer review checklist is included in Section 3.2. Any problems identified during the peer review are recorded and corrected before coding begins (see Section 3.3 for Metrics collection).

When the peer review is completed, and problems resolved, the reviewer completes the peer review section of the CCR.

Code

After the reviewer signs off on the completed design review, the developer can begin coding. Coding standards for the languages used in CLASS are discussed in Section 4 of this guide.

The developer checks out the modules that will be modified, completes coding for new or modified modules, and ensures a clean compile. The developer may conduct some preliminary unit testing prior to the peer review, but should not complete full testing. Conducting testing prior to the review delays the review, and results in rework when testing needs to be repeated after problems are found during the review.

For large or complex functions that involve many new or modified modules, the developer may complete only a subset of the modules and schedule a peer review for that subset. The scope of an individual review should be small enough to be conducted in one or two hours, and large enough to include closely related modules. The peer review checklist is included in Section 3.2.

Any problems identified during the peer review are recorded and corrected before the reviewer signs off on the review (see Section 3.4 for Metrics collection). When the review is complete, and any problems resolved, the reviewer completes the code review section of the CCR.

Development Testing

Developers conduct two levels of testing for each CCR: unit testing of each new or changed module, and development integration of the completed CCR. The plan for unit testing is reviewed during the code review, and should specify any test routines and test data that are needed. The plan for development integration is reviewed during the design peer review.

The developer should complete as much testing as possible before committing the changes into the source code control system. Each night, the development system is rebuilt incorporating all new committed changes for that day. This nightly build provides the foundation for all development testing across the project. Developers should ensure that all code compiles and builds cleanly in their local environment before committing the changes to the controlled library. If the code checked in by a development team causes the nightly build to fail, the TAL will work with that team to help determine the cause of the failures and to improve the team's development practices.

The developer conducts final development testing after the nightly build has integrated the changes into the development configuration.

During the development test period, the developer also reviews any documentation related to the change and updates documentation as necessary, including completing development information on the CCR. Section 6 addresses documentation standards.

At the completion of development integration, a final peer review is conducted to verify successful completion of the testing and documentation. This review certifies that the software is ready for the independent integration and test team. The checklist for the test readiness peer review is included in Section 3.2. Any problems identified during the peer review are recorded and corrected before the software is turned over (see Section 3.4 for Metrics collection).

When the test readiness peer review is complete, the developer notifies the software development lead that the software is ready for turnover. The developer changes the CCR Status to Pending – Supervisor Action, and the Supervisor to the Software Manager.

The software development lead verifies that all peer reviews have been completed and all metrics have been captured, and reassigns the CCR to the CMO for promotion to Integration.

3.2 Peer review checklists

The checklists for each of the peer reviews for the development process are provided below.

Design Review Checklist

Design Review Checklist		
By signing this Review form, the assigned Verifiers are indicating that the product has been reviewed for the following:		
X	**Review Criteria**	**Comments**
	1. The documentation conforms to documentation standards.	
	2. The Product Requirements (or Project Definition) are satisfied where applicable to Product Design.	
	3. Consideration was given to reliability and maintainability.	
	4. Due consideration was taken of experience from previous developments.	
	5. The design can be implemented with the available technology.	
	6. The design has been validated for the required functionality.	
	7. The design includes or references acceptance criteria.	
	8. The design conforms to the appropriate statutory and regulatory requirements.	
	9. The design conforms to the CLASS design standards.	
	10. Test procedures and data have been defined for integration testing.	

If the development team (developer and reviewer) identifies exceptions to the above checklist for a specific review, the reviewer documents the exceptions in the comment section, indicating waiver required. Waivers must be approved by the responsible development group lead and the lead integrator.

Code Review Checklist

Code Review Checklist		
By signing this Review form, the assigned Verifiers are indicating that the product has been reviewed for the following:		
X	**Review Criteria**	**Comments**
	1. The code implements the requirements stated in the CCR	
	2. Code accurately represents the detailed design as documented	
	3. There is adequate error checking and message logging for changes made in this CCR	
	4. Files and database tables are accessed correctly	
	5. Redundant and unexecutable code is avoided	
	6. Nested loops beyond six levels are avoided	
	7. Use of variable names, terminology, and identifiers follows CLASS standard	
	8. Environment variables are adequately defined. Hard coded quantities are avoided in every code unit in the CCR	
	9. Use of practices that bind the code to unique device characteristics is avoided (platform / hardware independent)	
	10. Every code unit is sufficiently readable and self explained. Comments are adequately provided	
	11. Every file and function has a prolog correctly formatted for automatic extraction into reference documentation.	
	12. Every file has a version control identifier.	
	13. Every code unit compiles cleanly	
	14. The CCR test plan is valid for the current release	
	15. The developer has provided a complete list of code units for this CCR	
	16. The developer has accounted for all environment files, and test data needed to test	
	17. The developer has a valid plan for testing the changes made under this CCR	
	18. The developer has clear instructions for build	
	19. The developer has clear instructions for testing	

If the development team (developer and reviewer) identifies exceptions to the above checklist for a specific review, the reviewer documents the exceptions in the comment section, indicating waiver required. Waivers must be approved by the responsible development group lead and the lead integrator.

Test Readiness Review Checklist

Test Readiness Review Checklist		
By signing this Review form, the assigned Verifiers are indicating that the product has been reviewed for the following:		
X	**Review Criteria**	**Comments**
	1. Have all outstanding items from previous reviews been resolved?	
	2. Was required testing successfully completed?	
	3. Is the CCR complete?	
	4. Is all related software ready for promotion?	
	5. Was a representative set of tests successfully run under peer review?	
	6. Does the delivery contain the required data files?	
	7. Does the delivery include any needed special build or operating instructions?	
	8. Have all the resolved change requests been identified in the delivery?	
	9. Is there a complete, correct, and clear identification of any outstanding problem reports?	
	10. Is all required documentation ready for delivery?	
	11. Does the user documentation clearly and completely describe how to use the software in its operational environment?	
	12. Have security requirements been met?	

If the development team (developer and reviewer) identifies exceptions to the above checklist for a specific review, the reviewer documents the exceptions in the comment section, indicating waiver required. Waivers must be approved by the responsible development group lead and the lead integrator.

3.3 Metrics collection

At turnover, each TAL updates the Process Measurement Form (see Attachment A) to record the results of the peer reviews. Additionally, the TAL updates the release notebook with actual software size and effort data.

4 Design & Coding standards

This section provides design and coding standards for CLASS development, for the most commonly used programming languages. Use of any language other than those listed here must be approved by the SET, and coding standards documented.

4.1 Design goals

Maintainable software and *simple operation* are the basic design goals for CLASS. Some specific goals are listed below and the design and implementation approaches that have been adopted to meet these goals are discussed.

Easy addition of new data types

- Software is general and parameterized. Parameters that define file and record structures for ingest purposes are contained in the database, so that the same software can be used to process many different types of files. Likewise, the content and appearance of the web pages in the user interface are largely controlled by parameters stored in the database, so that new data types and search criteria can be added easily.
- Application software is object-oriented. While some software, particularly in the Ingest system, is specific for certain data types, there are many common and reusable classes that facilitate the creation of new specialized classes through composition and inheritance.

Easy addition of new functions

- The application architecture is highly modular. CLASS is divided into major components (e.g., SAA), which are divided into subsystems. The more complex subsystems, such as Ingest, Recall, and Delivery, each consists of several independent processes supervised by the Activity Control system. Independence means that each process performs a function based solely on the contents of the item-descriptor that it receives, with no knowledge of the other processes that have handled or that will handle that item-descriptor. This approach enables the implementation of new functionality in new processes with minimal impact to existing code.
- The Activity Control system supports the easy modification of processing paths and the addition of new processes. Types of items to be processed (e.g., data sets, orders, line items) and processing paths (activities and triggers) are defined in database tables that can be easily modified.
- The availability of utility classes facilitates the development of new processes. An Activity Control client class provides methods that enable any process to obtain item-descriptors in priority order, update activity status, and re-queue item-descriptors. There are classes that perform common functions such as querying and updating database tables, creating log messages in standard formats, and transferring files.
- Adherence to a standard design facilitates the development of new processes. All transient processes that run under the Activity Control system have the same high-level design, essentially:

```
Initialize activity logging
Open database connection
DO WHILE there are items to process
   Get next item from queue
   Process item
   IF error
      Write error message to log
      Put item back in queue for later re-processing
   ELSE
      Let ActivityController know that this activity is completed
   ENDIF
ENDDO
```

There are utility classes that are tailored to perform the common functions within this design pattern.

Automatic processing and recovery

- The ProcessStarter (a component of the Activity Control System) automatically starts processes when they are needed. The ProcessStarter itself is periodically restarted by cron to ensure continuous operation.
- The Activity Control System automatically restarts failed processes
- The Activity Control System automatically re-queues items that may have been incompletely processed because of system failure
- Software is designed to recover automatically when resources are temporarily unavailable. The Activity Control system maintains a queue of item-descriptors waiting for each process. If a process cannot complete an action on an item because some resource is unavailable, e.g., a file system is filled up, that process returns the item to the queue, initiates cleanup, and periodically attempts to complete the failed action. Thus the unavailability of disk space may bring a process effectively to a halt, and this may cause other file systems to fill up and bring other processes to a halt, but item-descriptors remain queued and all processing resumes automatically once the root problem is resolved.

Standard activity and error reporting

- Application processes write activity and error messages in standard formats to a set of log files. A Log Monitor program periodically reads the log files and sends messages of specified types via e-mail to designated operators.
- The Activity Control System monitors the activity of each process and the progress of each item through the system. It issues operator alerts (via the log file and Log Monitor mechanism) for any process that appears to be inactive or slow, or any item that appears to be stalled at some point in its processing path.

Centralized monitoring and control

- The Activity Control system supports centralized control of distributed processing. Process parameters (run permissions, hosts, number of instances of

each process) are defined in database tables that operators can change to halt or resume processing at any point or to reconfigure the system.
- A web browser operator interface enables operators to monitor processing, restart or cancel orders, modify runtime parameters in the database, and control processing through the Activity Control tables.

4.2 Coding standards

CLASS follows the coding standards defined in the Software Standards for Information Processing Division (IPD). The following programming languages are used in CLASS: C++, Java, Perl, and JavaScript. Use of any other language must be approved by the SET.

5 Testing approach

This section briefly describes the overall testing approach. More details on CLASS integration and testing are included in the CLASS Test Plan and Test Procedures.

5.1 Levels of testing

Testing for CLASS can be categorized into two areas: development testing and independent system integration and test.

For development testing, the developer of a software change conducts unit level testing and integration of the modules related to the specific change. This testing does not usually require formal test plans and procedures, although the general planned test approach is reviewed during the design and code peer reviews, as discussed in Section 3. At the discretion of the software development lead, for more complex or critical functions, a formal plan and procedures may be required. In that case, a peer review of the plan and procedures will also be conducted.

An independent system integration and test team conducts formal integration testing of all software changes after the developer has completed development testing and the software development lead has approved the software for turnover to the test team. These tests are conducted according to the approved test plan and procedures, as allocated to the release, and include

- System build and verification in the Integration environment
- Promotion to the Beta Test environment for functional and stress testing of new functionality, and regression testing
- Promotion to the NCDC Test environment for deployment testing and NCDC readiness testing

At the successful completion of system integration and testing, the Lead Integrator approves the release and notifies the CPMT that the software is ready for promotion to operations. The CPMT makes the final approval of promotion to operations. With CPMT approval, the CMO promotes the system to the Operational environment, as defined in the CM Plan.

5.2 Test documentation

The system test procedures to be followed for a release are identified in the test procedures documentation. The test identification is entered into the release test report, along with the results of each test. In the event that a function fails a test, the failure is recorded as well as results of subsequent re-tests. This test report is included in the Release Notebook at the conclusion of testing.

5.3 Problem tracking

Problems encountered during development testing are corrected by the developer as they are identified and do not need to be recorded.

Problems encountered by the system integration and test team are recorded as Problem Reports (PRs), and the software development lead is notified of the need for correction. The status of PRs is tracked during the System Integration and Test phase: the CMO prepares weekly (or on request) reports for the Lead Integrator and development team leads. As each PR is corrected, the test team retests the function.

If the CPMT approves, based on the severity of the problem, it is possible to promote the new release to operations with outstanding unresolved PRs. In that case, the PRs are converted to CCRs, since they now represent a requested change to the new operational baseline.

5.4 Metrics collection

At the completion of system testing, each development group updates the Process Measurement Form (see Attachment A) to record the results of the test phase. Additionally, the TALs update the Release Notebook with updated software size and effort data, and the test report.

6 Software documentation standards

This section describes the standards for software design documentation and CCRs and PRs. Standards for documenting the source code are included in the Software Standards for IPD.

6.1 Software Design Documentation

The following documentation outline is the standard format for CLASS design documentation. Sections not relevant to a particular design should be included and marked as N/A. Additional information should be added as necessary to convey an adequate understanding of the design. If a different format is more useful to conveying a complete picture of the changes being made and the impact to the other system components, the developer should obtain approval from her or his development lead for a revised format.

After the design is implemented, much of the overview documentation presented here,

including diagrams, will be incorporated into the overview sections of the software description documents in the online CLASS library. The design details should be reflected in the in-line comments and source file prologs that are extracted to produce the software reference documentation. Descriptions of database table changes will be incorporated into the database documentation.

1. Requirements

State the requirements that are driving this enhancement or change

2. Subsystem Design Overview

Discuss the subsystem design or design changes at a high level. Include, as appropriate:

- Subsystem processes affected - what is the function of each process and how is that functionality being altered
- Database changes
- New or modified user interface pages
- New or modified interfaces

3. Activity Control Paths

Define new paths or path modifications for processes that run under the Activity Control System.

4. Parameter and Configuration Files

Describe new files or modifications to existing files that are used in operations, other than executable programs. For example: site maps, style sheets, XSP files, environment files, e-mail template files. If pipeline definitions in the site map are affected, discuss and illustrate each new or modified pipeline.

5. Storage Areas

Identify new permanent or temporary storage areas required. Estimate space requirements. Indicate any special maintenance procedures (e.g., cache cleanup procedures).

6. Log Message

Give examples of new log messages to be generated. Identify log directory in which these messages will be kept.

7. Interprocess Communication

Describe formats of new or modified interprocess messages, search results files, visualization product files.

8. Database Tables

Describe additions or changes to the structure and content of the database:

- *Structure* - Describe new tables and modifications to existing tables. Identify column names, variable types, contents of each column
- *Content* - Describe new sets of parameters to be added, e.g., to define new activity control paths or to ingest new data types. Identify all tables affected.

9. Program Design

Provide the following for each affected process in the subsystem:

- Process functions
- Diagrams - Include whatever diagrams might be useful in clarifying the workings of the process, e.g., class association diagrams, state transition diagrams, activity diagrams
- Class design - Identify major attributes or methods that are being added or modified. Describe the input and output for each method. Describe the function of each method, using PDL for complex functions.

10. Operational Characteristics

Discuss ways in which operators may monitor and control the new or modified system. Discuss the circumstances under which an operator may be required to intervene to resolve problems.

11. Test Plans

Discuss plans for testing the changes. Identify special environments, tools, or data that will be required for testing.

12. Requirement Mapping

Map requirements to design, i.e., provide a table in which each requirement is mapped to the above sections that describe design elements driven by that requirement.

6.2 Software Description

The software description section of the online library describes the software as built. The documentation of a subsystem is generally formatted as follows:

- Functions and Interfaces: High level description of functions and interfaces; context diagrams
- Design: Overview of subsystem design; separate subsections for overviews of major components:
  - Program A
  - Program B
  - Etc.
- Special Interfaces and Formats: Formats of external files, messages, data structures
- Implementation: Locations of source and runtime files; build procedures; programmer Notes
- Operation: Required environment; notes on execution and monitoring
- Diagrams: Class Association Diagram
- Reference Documents
  - C++ Class Hierarchy
  - C/C++ Classes, Structs, Unions
  - C/C++ Class and Struct Members
  - C/C++ File List

- C/C++ File Members
- Script List
- Java Class Hierarchy
- Java Class Fields and Methods
- Java Package Index

All the Sections except Reference Documents shown in the above outline contain overview information that must be supplied by the software developers. Some sections (e.g., Formats, Operation, Diagrams) may be omitted if not applicable.

The Diagrams section of the document contains links to Tau-UML diagrams and other diagrams, preferably in PostScript format.

The Reference Documents section contains only information generated by the tools doxygen, scriptdocgen, and javadoc.

The locations of the online document files, tools available for generating reference documentation, and the procedures for updating the Software Description documentation are presented in the online library under Software Documentation Standards and Tools.

6.3 Configuration Change Requests

Figure 6 shows the standard Remedy tool CCR form. The CCR originator fills out the initial part of the CCR when he or she submits the change request. The developer completes the description of the change when the change is implemented:

- Each file that is modified must be noted in the CCR along with the version of that file. This is done automatically by CVS when the file is committed. The list of runtime files that will be moved into the operational environment must be added to the CCR. This information goes in the Runtime Files field of the form.

- In the Instructions field of the form the developer should include:
  - Build instructions
  - Customizations required
  - Any other instruction that the developer considers necessary.

- The Developer documents all changes done in the Implementation field of the Form.

- The Developer writes a test plan and updates the Test field with it. Expected test results should also be provided.

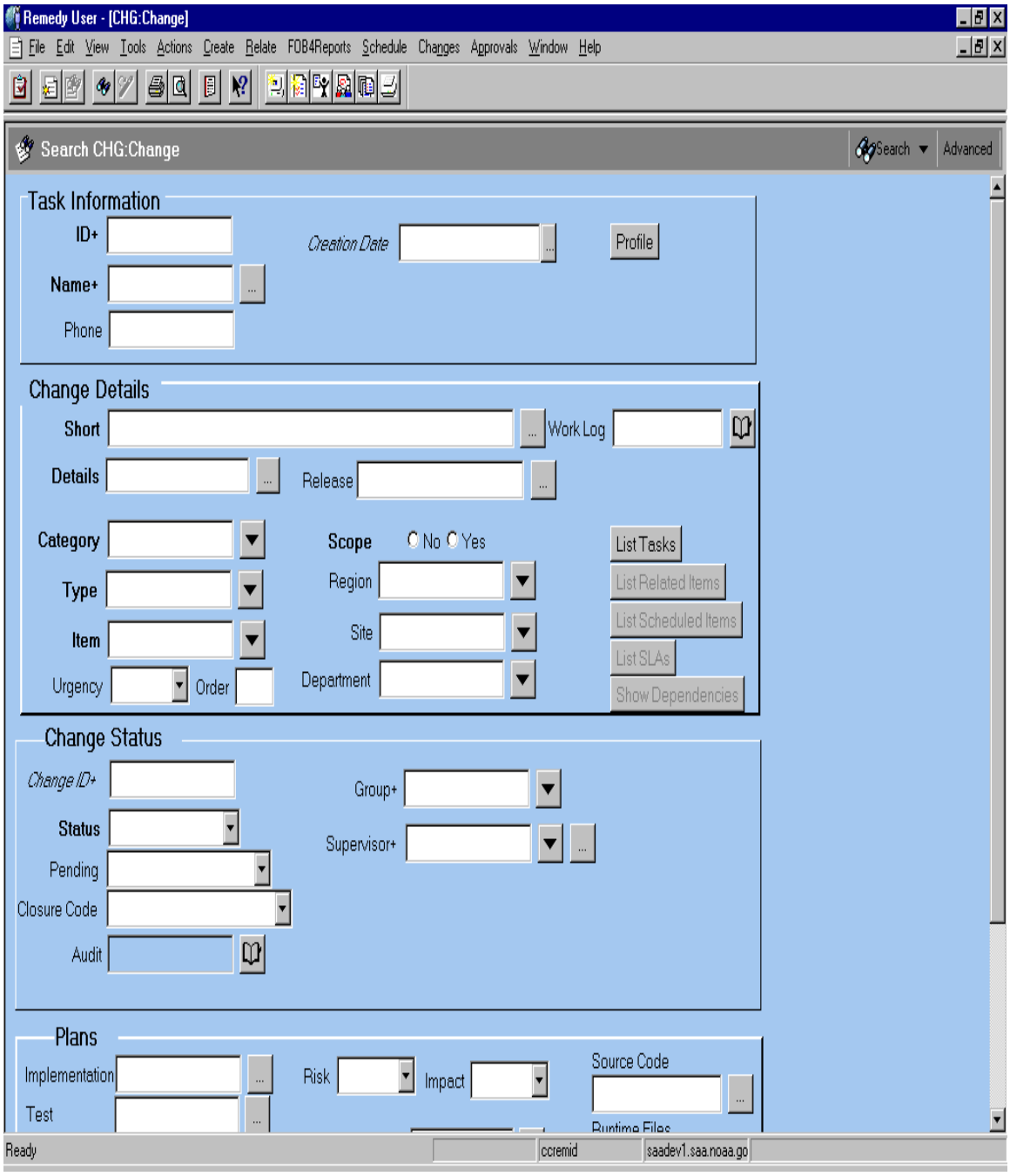


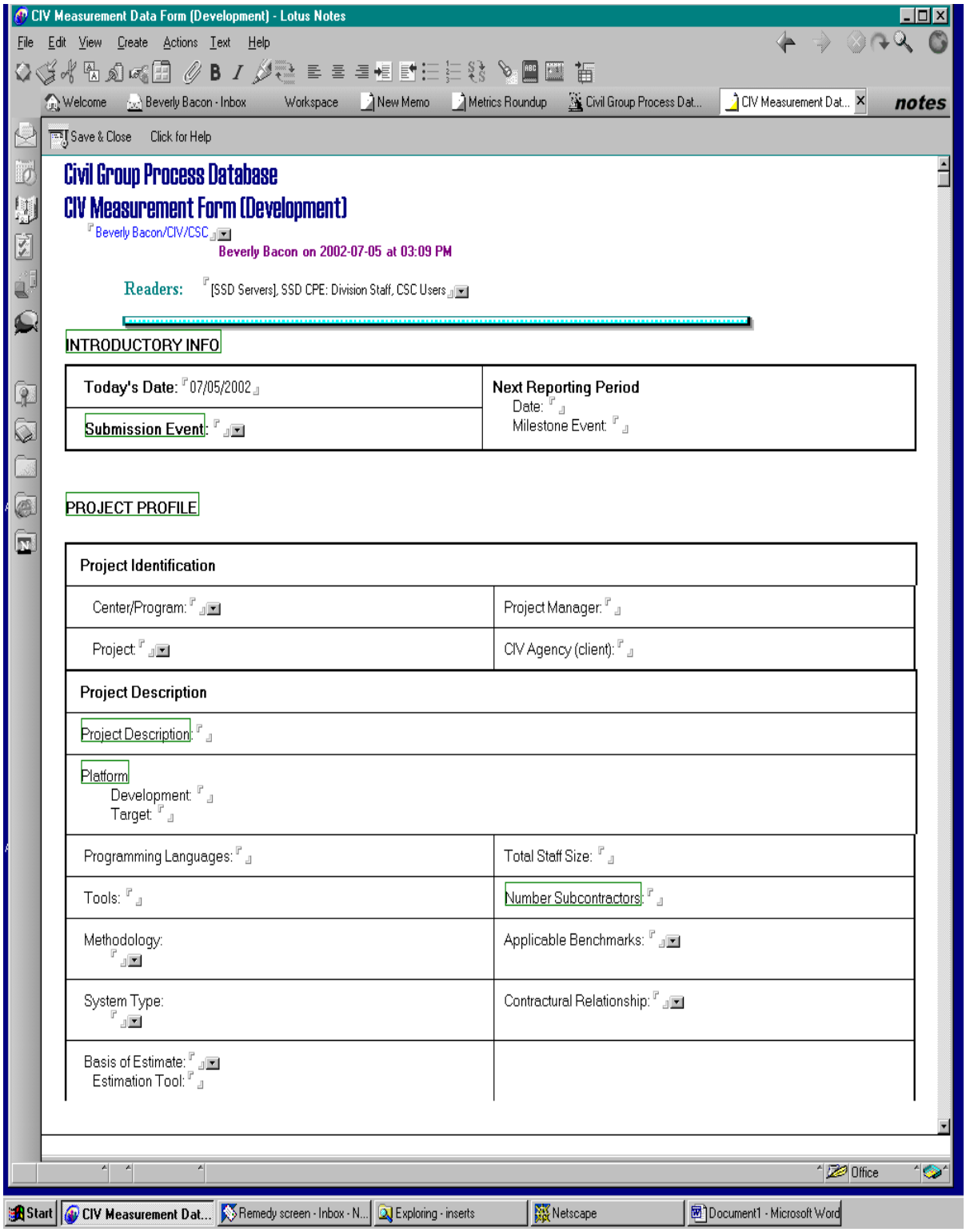
Figure 6 - Configuration Change Request Form

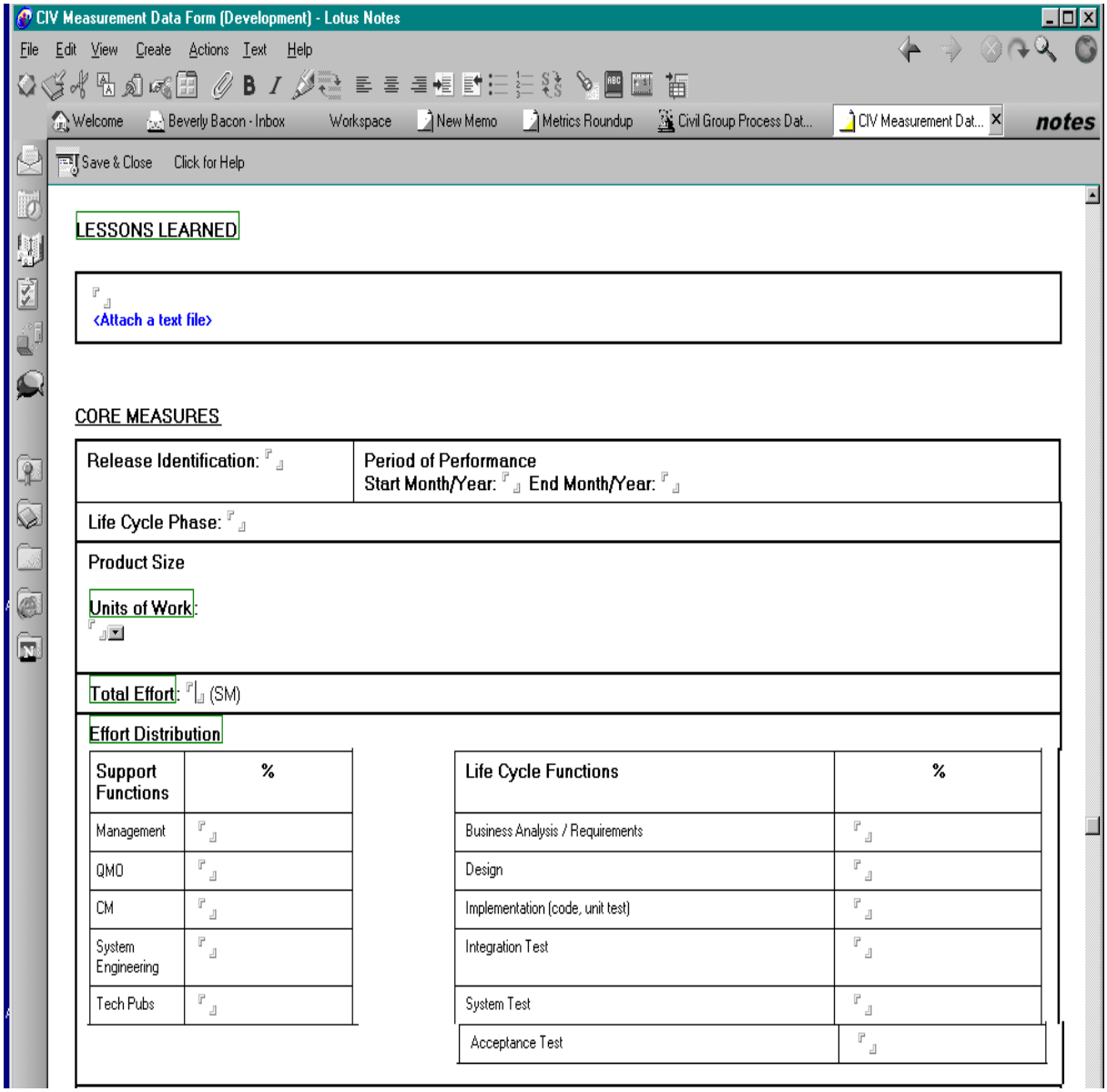
The CLASS Configuration Management Plan provides additional details on the use of the Remedy CCR tool.

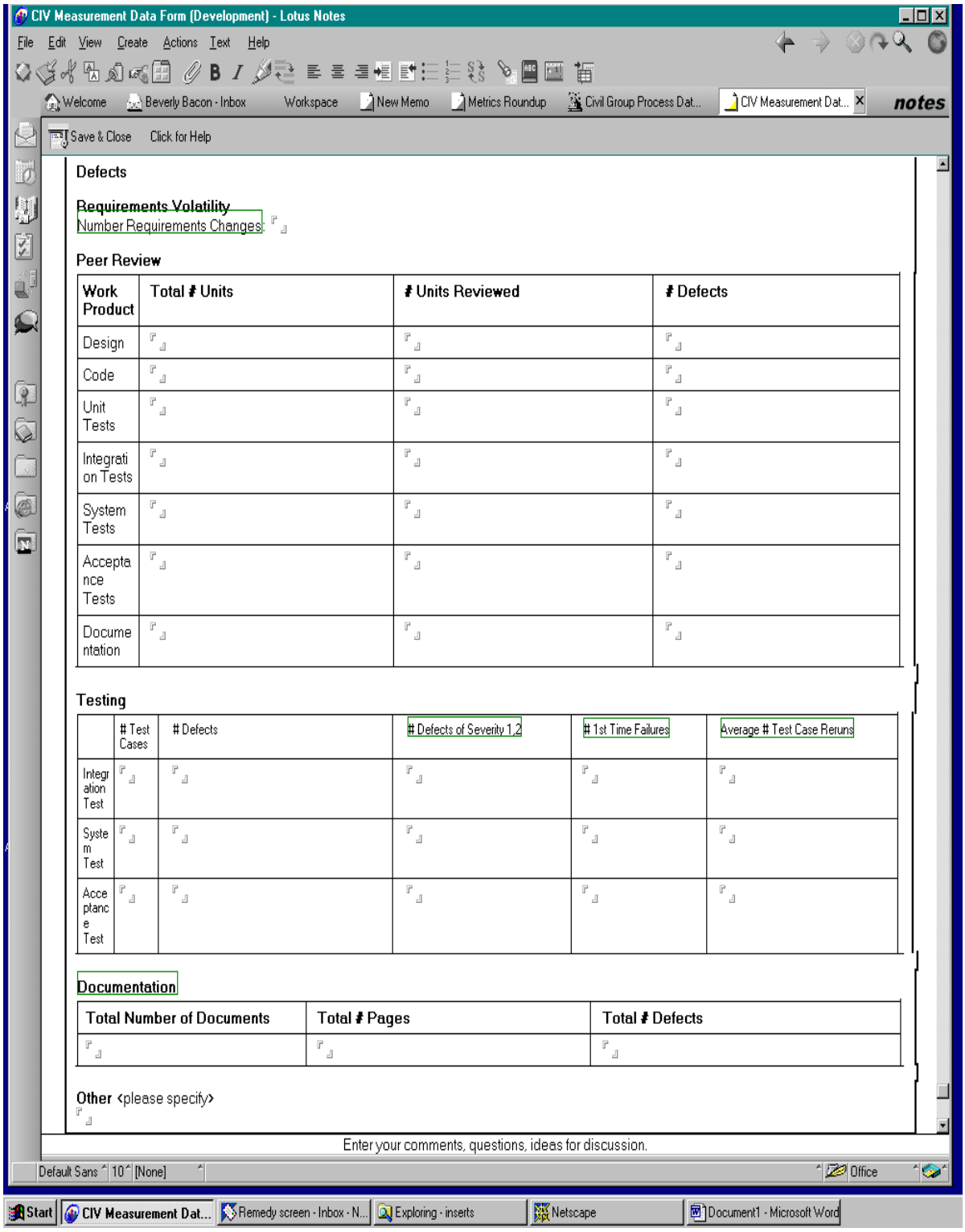
6.4 Problem Reports

The Problem Reports use the same tool and form as the CCRs, and are distinguished by the PR indicator only. As in the CCR, the developer must identify each file that is new or changed in correcting the problem.

Attachment A – Process Measurement Form




CIV Measurement Data Form (Development) - Lotus Notes
File Edit View Create Actions Text Help
Welcome Beverly Bacon - Inbox Workspace New Memo Metrics Roundup Civil Group Process Dat... CIV Measurement Dat... notes
Save & Close Click for Help
LESSONS LEARNED
<Attach a text file>
CORE MEASURES
Release Identification:
Period of Performance
Start Month/Year: End Month/Year:
Life Cycle Phase:
Product Size
Units of Work:
Total Effort: (SM)
Effort Distribution
Support Functions
%
Management
QMO
CM
System Engineering
Tech Pubs
Life Cycle Functions
%
Business Analysis / Requirements
Design
Implementation (code, unit test)
Integration Test
System Test
Acceptance Test


CIV Measurement Data Form (Development) - Lotus Notes
File Edit View Create Actions Text Help
Welcome Beverly Bacon - Inbox Workspace New Memo Metrics Roundup Civil Group Process Dat... CIV Measurement Dat... notes
Save & Close Click for Help
Defects
Requirements Volatility
Number Requirements Changes:
Peer Review
Work Product | Total # Units | # Units Reviewed | # Defects
Design
Code
Unit Tests
Integration Tests
System Tests
Acceptance Tests
Documentation
Testing
Test Cases | # Defects | # Defects of Severity 1,2 | # 1st Time Failures | Average # Test Case Reruns
Integration Test
System Test
Acceptance Test
Documentation
Total Number of Documents | Total # Pages | Total # Defects
Other <please specify>
Enter your comments, questions, ideas for discussion.
Default Sans 10 [None] Office
Start CIV Measurement Dat... Remedy screen - Inbox - N... Exploring - inserts Netscape Document1 - Microsoft Word

Attachment B – Peer Review Procedure

1. OWNER

This procedure is derived from Computer Sciences Corporation (CSC) Civil Group (CIV) defect prevention procedures, for use on the CLASS project. Any recommendations for improvement should be submitted to the CLASS Quality Management Office.

2. PURPOSE

This procedure defines the peer review and work product inspection (WPI) process, including methods, roles, and responsibilities. Peer Reviews and WPIs are scheduled and conducted throughout the system development life cycle. The purpose of these reviews and inspections is to remove defects from the work products early and efficiently. The reviews and inspections involve a methodical examination of the work product by the author's peers to identify defects and areas where changes are needed.

3. RESPONSIBILITIES

3.1 Development group lead

The development group lead is responsible for defining the level of review required for each element of a release (e.g., Peer Review or WPI), and assigning an individual or team to conduct the review. The development group lead also arbitrates any conflict between the developer and reviewer.

3.2 Quality management

Quality management (QM) personnel are responsible for ensuring that the Peer Review and WPI process are conducted in accordance with the project's documented procedures and standards. QM is also responsible for periodically auditing the project data repository to ensure the inclusion of metric data associated with these reviews/inspections and to verify closure of Peer Review and WPI action items.

3.3 Project Technical Personnel

These personnel are knowledgeable in the objectives, principles, and methods of the Peer Review and WPI process as well as their assigned roles in the process. These roles include

- Facilitator—Leader of the review responsible for managing all aspects of the review meeting. The facilitator prepares the objectives of the review, notifies the participants about the review schedule, assigns a recorder, oversees the orderly conduct of the review, ensures that review minutes are prepared and distributed,

and provides defect and action item follow-up to ensure closure in a timely fashion.

- Recorder—Responsible for documenting all discrepancies and defects found, suggested improvements, and assigned action items.
- Author—Responsible for generating the material being reviewed and implementing changes to the material as required.
- Reviewers—Cognizant representatives of author's peer group, which can include project personnel from Systems Engineering, Software Engineering, QM, or Software Test. Reviewers are responsible for technical review of the material and feedback on all defects discovered.

4. INPUT

Input for Peer Reviews and WPIs is as follows:

- Project standards and checklists for the product being reviewed/inspected
- Peer Review/inspection process
- Project plans and schedules:
  - Program Management Plan (PMP)
  - Project QM Plan
  - Review/inspection schedule
- Product requirements and acceptance criteria
- Product(s) to be reviewed/inspected

5. PROCESS

Any type of technical or management product may be reviewed or inspected, whether it is deliverable to the customer or internal to the project and whether it is in an intermediate or final state. The goal of the Peer Reviews and WPIs is to find and correct errors as early in the system life cycle as possible. These reviews and inspections are interactive, with a focus on constructive criticism. Peer Reviews and WPIs focus on the work product being reviewed and not on the author of the product.

PRs are normally conducted to certify products before proceeding to the next phase of development. PRs may involve only two people or a group. WPIs are structured walkthroughs used to verify correctness and completeness of products or processes. Inspections are typically used when feedback is required from cross-functional or cross-organizational groups, when the scope of the feedback is relatively large, when the product or process is relatively complex or has a far-reaching impact, or when education of a group is necessary. In WPIs, the manager of the work product being inspected may participate in a technical capacity, but the manager does not use the results of the review or inspection to evaluate the performance of the work product author.

The specific products that undergo PR and WPI are identified in the project's defined process and scheduled as part of the project planning activities.

Common features of the PR and WPI process include

- A triggering event defined within the authorizing document, or a request from functional management, QM, or the developers
- Identification of the review participants and their roles in the review
- An established schedule, with notification sent to affected personnel
- A completed product to be reviewed or inspected according to the established requirements and standards
- The actual review or inspection, with action items assigned for any unresolved issues or questions identified during the review; all action items have a designated assignee and required completion date

The specific processes associated with the one-on-one and group PRs and the WPIs are described in the following subsections. Table 5-1 identifies the review attributes associated with these three types of reviews and provides selection criteria guidance.

5.1 One-on-One Peer review

A one-on-one PR involves only two people and is intended to provide feedback on identified defects in a product. The reviewer is normally a co-worker of the product author who is knowledgeable in the area being reviewed. The author's manager is not usually the reviewer, as the review is intended to evaluate the product, not the author. One-on-one PRs are appropriate in situations where the review attributes meet the criteria shown in Table 5–1.

Table 5–1. PR and WPI Selection Criteria

Review Type / Review Attribute	One-on-One PR	Group PR	WPI
Provide feedback to developer(s) early in the development cycle (e.g., prototype material)		x	x
Provide immediate feedback to developer(s) (e.g., unit detail design)	x		
Individual feedback	x		
Group feedback		x	x
Material simple, straightforward	x	x	
Material complex, technically risky		x	x
Review scope small (e.g., single unit)	x	x	
Review scope large (e.g., system interfaces)		x	x
Customer involved			x
Determine alternate approaches		x	x
Minimal resources	x		
Predistribution of material required	x	x	

Review Type / Review Attribute	One-on-One PR	Group PR	WPI
Preparation required	x	x	
Product certification	x	x	
Product verification		x	x
Product maturity check		x	x
Product progress check		x	x
May require multiple reviews			x
Precursor to formal review		x	x

5.1.1 Preparing a One-on-One Peer Review

One of the first steps in a one-on-one PR is to identify the reviewer. The manager or technical lead may make the assignment, or the author may ask a co-worker on the project to be the reviewer. If the project has very few members, such as one or two people, the reviewer may be from a similar project rather than from the same project.

Once the reviewer has been identified, the author and reviewer establish a schedule for the review process. The schedule may be determined by a set turnaround time or a milestone date. The author provides the review material to the reviewer.

5.1.2 Conducting a One-on-One Peer Review

The reviewer may look at the material alone or with the author. The reviewer evaluates the material against any certification criteria (often expressed with a checklist) defined for that product type as guidance for evaluating the product. A checklist provides evaluation guidance for indicators of quality, helping the reviewer to focus on the types of errors likely to occur with a particular type of product.

Any review comments are returned to the author, who then revises the product. The rework is then reviewed by the reviewer. This continues until the reviewer is satisfied with the product and can certify it. If the author and reviewer do not agree on the identification of a problem or how to resolve it, it may be brought to a third person for resolution or documented as an action item. The inability to come to consensus may indicate that a one-on-one review is inappropriate for this product.

The reviewer may determine that the changes being reviewed are extensive or complex enough to require the group PR method. In that case, the one-on-one PR is ended, and the group PR is scheduled.

5.1 3 One-on-One PR Follow-Up and Products

The output of the review is a certification that the product or set of products has successfully passed the PR criteria. The certification is available as an audit trail. In addition, any errors and issues that were outside the scope of the review are logged for later corrective action.

5.2 Group Peer review

Group PRs consist of the product author(s), a group facilitator, and the reviewers, who are peers of the person(s) whose product(s) is being reviewed. Group PRs are appropriate in situations where the review attributes meet the criteria shown in Table 5–1. Group PRs consist of three phases: preparation, conduct, and post-review. The following subsections describe the activities associated with each of these phases, and Table 5–2 summarizes the roles and responsibilities by phase.

5.2.1 Preparing a Group Peer Review

During the preparation phase, the author(s) of the material to be reviewed ensures that the material is complete, obtains management approval to schedule the review, and informs the facilitator that the material is ready for review. After receiving notification, the facilitator identifies the review team, schedules the review, prepares a package of review material, and notifies participants. Review material is distributed to reviewers in advance of the PR. The review team reviews the product distributed to them and prepares their questions and concerns prior to the review.

5.2.2 Conducting a Group Peer Review

The second phase in the PR process is the conduct of the meeting. The function of the meeting is to identify and record defects found by the reviewers during their independent review preparation. The PR meeting is highly structured. The main focus is the identification, not the correction, of defects.

Table 5-2. Group PR Roles and Responsibilities

Role / Phase	Facilitator	Author(s)	Reviewers
Preparation	• Verify review material readiness • Schedule meeting • Distribute list of review material • Notify participants of meeting time/place • Notify author if additional material is required • Notify participants if meeting must be rescheduled • Conduct technical inspection of material submitted for review	• Notify lead engineer that material is ready for review • Provide list of material to facilitator • Provide any additional material needed by facilitator	• Conduct technical inspection of material before meeting using recommended review checklist(s) • Alert facilitator of additional material needed to supplement review • Document review findings, comments, questions, and suggestions • Record all technical review time

Role / Phase	Facilitator	Author(s)	Reviewers
Conduct	• Chair the PR meeting • Participate in meeting discussions • Keep meeting focus on immediate task • Ensure that all concerns are discussed and recorded (an additional review participant can be designated as meeting recorder) • Review defects list • Determine whether a reinspection is necessary	• Participate in meeting discussions	• Present all defects, questions, and concerns discovered during early review of the material • Participate in meeting discussions
Post-Review	• Generate and distribute meeting minutes, defects, and action items list • Monitor status of defects generated in review • Generate final defects report upon resolution of defects • Gather, record, and update PR project metrics	• Resolve all defects identified • Resolve any action items assigned	• Resolve any action items assigned

During the PR itself,

- An overview of the products under examination is presented (by the author[s] or the facilitator).
- Each product is reviewed in detail to allow discussion among the review team.
- The facilitator encourages questions from the review team to ensure all issues and concerns become visible and are elaborated upon.
- The facilitator, or a recorder appointed by the facilitator, ensures that all the deficiencies, issues, and suggested improvements are correctly noted and documented findings are distributed within a reasonable time frame to the complete review team.

The defects and action items are recorded if there is a consensus among the participants that a defect has been identified. All defect resolutions are the responsibility of the author of the material.

5.2.3 Peer Review Follow-Up and Products

In the final phase of the PR process, the facilitator produces and distributes the review documentation, including meeting minutes, defects recorded, and action items lists. The author(s) resolves all defects recorded during the conduct phase and any action items assigned. During this phase, the facilitator also ensures that the action items and defects assigned are completed in a timely manner. After all defects and action items are resolved, the facilitator distributes a final defect/action item report with the change in disposition status.

5.3 Work Product Inspection

WPIs are used when feedback is needed from a large group. This inspection is conducted as a structured walkthrough for certification review by peers, managers, or the customer and for educating others. WPIs are appropriate in situations where the review attributes meet the criteria shown in Table 5–1. WPIs consist of three phases: preparation, conduct, and post-review. The following subsections describe the activities associated with each of these phases.

5.3.1 Preparing a WPI

During the preparation phase, the author(s) of the material to be reviewed ensures that the material is complete, obtains management approval to schedule a review, and informs the facilitator that the material is ready for review. When changed material is being reviewed, the author gives the rationale for the change (such as providing a problem report number). After receiving notification, the facilitator identifies the review team, schedules the review, and notifies participants.

5.3.2 Conducting a WPI

At the inspection, the author(s) provides a detailed presentation of a product or portion of a product. The author walks the reviewers through the product to review its contents, discuss its details, and identify errors and issues. The review material is generally provided at the time of the walkthrough, though it may be provided in advance. Defects identified and issues raised at the inspection are recorded for resolution outside the review.

If the inspection is used to certify a product, then the product is also reworked after the walkthrough to address the errors and issues. Errors and issues that are outside the scope of the review do not need to be resolved, only logged, to grant the certification.

5.3.3 WPI Follow-Up and Products

If the inspection is used to certify a product, the defects identified are reworked by the author(s) and the rework is examined by the facilitator. This continues until the facilitator is satisfied with the product and certifies it. The output of the review is certification that the product or set of products is ready for the next phase of the life cycle. The certification is available as an audit trail.

If the inspection is used for verification of completeness and correctness, the facilitator produces and distributes the meeting minutes, defects recorded, and action item lists. The author(s) resolves all defects recorded during the conduct phase and any action items assigned. During this phase, the facilitator also ensures that the action items and defects assigned are completed in a timely manner. After all defects and action items are resolved, the facilitator distributes a final defect/action item report with the change in disposition status.

With both kinds of inspections, any errors and issues that are outside the scope of the review are logged for later corrective action.

6. OUTPUT

The output of one-on-one PRs includes the updated product with identified problems resolved and a certification record in the project's certification repository. The output of the group PR and WPI process is minutes documenting the deficiencies, issues, suggested improvements, and action items. These minutes should include

- Names of attendees
- Meeting duration time
- Amount of material reviewed
- List of defects and/or action items
- Total number of defects
- Number of major defects
- Number of minor defects
- Total preparation time for review team
- Total preparation time for facilitator
- Disposition of elements reviewed (e.g., accept, verify rework, reinspect)

Additionally, the project data repository must be updated to account for the conduct of the review and for the resources expended performing the review.

A PR or WPI is complete when all open defects and actions items that have been documented are assigned a resolution type of anything other than "open."

7. TAILORING

- Number of participants – PRs and WPIs should be limited to a small number of participants. Typically, a review team includes three to seven subject matter experts and one facilitator.
- Review scope and meeting duration – PRs and WPIs should have an absolute time limit, typically no longer than 60 to 90 minutes. This time limit helps to determine the amount of material chosen for each review. If the meeting goes over the allocated time, follow-up meetings should be scheduled.
- Checklists or certification criteria – For PRs and when the inspection is used for certification, the standard against which the product is being evaluated must be established. How and where (e.g., software development folders) the certification is to be recorded must also be identified.
- Material re-review – Requirements for re-review should be set by each project and be based on the severity and number of defects found in the initial review of the material.

Attachment C – Estimation Worksheet

CLASS Estimation Worksheet

Estimating Unit Characteristics				
ID	**Estimating Unit**	**Complexity Productivity Estimate (hours/unit)**		
		Low	**Medium**	**High**

Work Estimate				
Partition ID	**Estimating Unit ID**	**Complexity**	**Size**	**Effort (Hours)**
Total Effort Estimate				

Attachment D – Acronyms

CCR	Configuration Change Request
CIO	Chief Information Office
CLASS	Comprehensive Large Array-data Stewardship System
CM	Configuration Management
CMO	Configuration Management Office
CPMT	CLASS Project Management Team
CVS	Concurrent Versions System
DoD	Department of Defense
EOS	Earth Observing System
GOES	NOAA Geostationary-orbiting Operational Environmental Satellite
IPD	Information Processing Division
Metop	European Meteorological Operational Satellite Program
NASA	National Aeronautics and Space Administration
NCDC	National Climatic Data Center
NESDIS	National Environmental Satellite, Data, and Information Service
NEXRAD	NOAA NEXt generation weather RADAR Program
NGDC	National Geophysical Data Center
NOAA	National Oceanic and Atmospheric Administration
NPOESS	National Polar-Orbiting Operational Environmental Satellite System
NPP	NPOESS Preparatory Program
OSDPD	Office of Satellite Data Processing Division
PAL	Project Area Lead
POES	NOAA and DoD Polar-orbiting Operational Environmental Satellites
PR	Problem Report
QM	Quality Management
SAA	Satellite Active Archive
SET	Systems Engineering Team (CLASS)
TAL	Technical Area Lead
WBS	Work Breakdown Structure
WPI	Work Product Inspection